CyberGAN: Generating High-fidelity Cybersecurity Data With Generative Adversarial Networks

Julia Gonik and Joie Le
Massachusetts Institute of Technology, Cambridge, MA, 02139

Arun Viswanathan
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

Yuening Zhang
Massachusetts Institute of Technology, Cambridge, MA, 02139

Machine learning for cyber defense offers the promise of detecting adversarial activity against the ground data systems managing critical space assets. A fundamental challenge facing machine learning research in cybersecurity is the lack of high-fidelity, shareable datasets for robust evaluation and testing of machine learning-based solutions. High-fidelity, real-world datasets are necessary for reliable benchmarking of nominal system behavior and malicious activity. Unfortunately, such realistic datasets of both nominal and adversarial activity are rarely shared publicly by data owners due to security and privacy concerns. Besides, the available adversarial data is sparse, which makes training models on malicious activity much harder. This situation has impeded and continues to impede the research and successful adoption of machine learning methods for cyber defense. Researchers have dealt with this problem by generating data within a low-fidelity lab environment, using classified and thus unshareable datasets, or downloading low-fidelity public datasets made available by others.

We propose an innovative solution to the problem by employing machine learning methods to generate high-fidelity data. Specifically, we propose the use of Generative Adversarial Networks (GANs) to generate high-fidelity data for cybersecurity purposes. GANs have found successful image processing and natural language applications, but have not yet been investigated for cyber data generation. Our proposed approach first involves training the 'discriminator' network of the GAN with a sample of real-world data consisting of malicious and nominal samples. We then use the 'generator' network to generate new high-fidelity data samples consisting of an appropriate mix of malicious and nominal activity. We demonstrate applications of our architecture by generating high-fidelity cybersecurity data containing both malicious and nominal samples. We thoroughly evaluate the fidelity of our generated data using heuristics and evaluate its usefulness for machine learning applications using three different datasets. Overall, our approach results in high-fidelity, shareable datasets.

I. Introduction

Machine learning for cybersecurity has the potential to revolutionize cyber defense techniques and detect adversarial threats against ground data systems managing critical space assets with great accuracy and reliability. High quality, abundant datasets are the cornerstone of most machine learning applications, and cybersecurity is no exception. Real-world data facilitate critical aspects of the machine learning pipeline for cyber defense, such as gauging nominal system behavior and adversarial interference. However, cybersecurity data are often limited in scope and accessibility due to security and privacy concerns. Furthermore, existing datasets can be sparse, hindering their potential utility. Despite these limitations, machine learning is becoming an increasingly popular tool in cybersecurity, and this surge in popularity necessitates the availability of large, reliable datasets for the training and evaluation of robust machine learning solutions. Nevertheless, the inability to obtain or generate high-fidelity cyber datasets impedes the development and acceptance of machine learning methods for cybersecurity and places significant limitations on the quality of available cybersecurity solutions.

Previous solutions to this data shortage included generating low-fidelity data within a lab environment, using publicly available datasets, or using classified and unshareable datasets. Using low-fidelity data, either available publicly or lab

generated, leads to inaccurate, non-generalizable, and non-deployable solutions. Using confidential datasets hampers research as researchers cannot effectively share their results with the broader scientific community. Further, others are unable to reproduce and validate the results, thus preventing the advancement of techniques that will be reliable and useful for real-world applications.

Due to the challenges posed by traditional methods of obtaining cybersecurity data, we turn to generative techniques to create high-fidelity synthetic datasets. In recent years, generative adversarial networks (GANs) [1] have garnered increasing attention as they display promising results in the fields of image and text generation [2–6]. A GAN is a class of deep neural networks that aims to produce synthetic data that retains the underlying distribution of the training data via competing generator and discriminator networks.

Despite the success of GANs in image and text generation, there has been substantially less research in the use of GANs for synthetic data generation. Xu et al. [7, 8] have previously utilized GANs for the generation of tabular data. GANs have also been applied for synthetic data generation in fields such as medicine [9] and energy [10], but little work has been done on generating synthetic cybersecurity datasets. Cyber datasets pose specific challenges that obstruct the use of generic data generation models. For instance, firewall records have unique relationships among fields or across records that capture the semantics of the computer network generating the data.

To address these challenges, we introduce CyberGAN, a novel GAN architecture explicitly designed for cybersecurity datasets such as firewall records. We utilize a simple, fully connected structure, domain-specific data processing, and a custom loss function to optimize our model for cyber data. When trained with existing cybersecurity datasets, CyberGAN performed significantly better than other GAN architectures.

Contributions We make the following contributions:

1) We introduce CyberGAN, a network that simplifies and outperforms previous architectures on mixed cybersecurity data with highly interrelated fields. The data generated by CyberGAN demonstrate robust performance on machine learning tasks in cybersecurity, such as the classification of nominal and adversarial firewall records.
2) We demonstrate how CyberGAN generates high-fidelity synthetic data (either anomalous or normal samples) that can be used as a substitute for or augmentation to existing cybersecurity datasets.
3) We rigorously evaluate our approach with three types of cybersecurity datasets: firewall logs, a credit card fraud dataset, and botnet network activity data.

The rest of the paper is structured as follows: Section II provides an overview of GANs, discusses related work and challenges for generating cybersecurity data. Section III discusses the architecture of CyberGAN. Section IV describes our rigorous evaluation methodology and metrics for evaluating the fidelity of generated datasets. Section V discusses the results, limitations and future work. Finally, Section VI concludes the paper.

II. Background and Related Work

This section first presents a brief overview of Generative Adversarial Networks (GANs), followed by related work on GAN-based data generation. We finally discuss challenges in generating cybersecurity data using GANs.

A. Overview of Generative Adversarial Networks

GANs initially evolved from the idea of pitting two algorithms against each other [1]. This rich class of generative models is characterized by two neural networks: a *generator* and a *discriminator*. Given real data from a training set, the generator produces synthetic samples fed downstream to the discriminator and classified as real or fake. These two networks essentially engage in a game of cat and mouse; the generator attempts to produce realistic enough data to trick the discriminator into making a mistake by classifying fake data as real.

The GAN training objective can be viewed as a two-player minimax game. The equilibrium state corresponds to when the generator outputs data from the target distribution, and the discriminator classifies any input it receives as "fake" or "real" with a 50% probability. It can be challenging to identify when this convergence has been reached. Prior work has developed methods to prevent issues such as *mode collapse* and improve stability during training, such as using the Wasserstein distance metric during optimization [11].

B. GAN-based Data Generation

In this section, we briefly summarize current work on generating different types of data using GANs.

Generating Images The majority of GAN research revolves around synthetic image generation. The most common datasets include MNIST (handwritten digits) [1] and CIFAR-10 (10 classes of objects). Some other notable applications of GANs include human face generation, image to image translation, text to image translation, face aging, and 3D object generation.

Success in the image domain, however, cannot be directly replicated in other domains. In the original GAN, training relies on backpropagation from the discriminator through the generated sample into the generator. The generator's gradients are obtained by differentiating a divergence computed by the discriminator with respect to the generator's parameters. For continuous data like images, this poses no issue. On the other hand, discrete data, such as text characters, are not differentiable and will result in vanishing gradients and backpropagation failure [3].

Generating Text Text-based GANs hold potential for data generation because of the incorporation of RNNs into the generator architecture. By combining a typical GAN structure with a recurrent network, these models effectively learn the distribution within and across training examples while generating new and varied text. However, since the selection of discrete text tokens during training is not differentiable, language GANs face unique challenges with optimization and backpropagation. In this work, we first explored the suitability of language GANs to learn the highly specialized structure of cybersecurity datasets. Many of these models [2–4] ultimately proved unnecessarily complex for generating structured cybersecurity data.

Generating Structured Data Our work builds upon and applies techniques from prior work on utilizing GANs for structured data generation. Several techniques for general-purpose synthetic data generation have been previously proposed, including architectures involving decision trees [12] and Bayesian networks [13, 14]. However, many previous methods are intended for specific distributions, such as medical records [9], or suffer from computational limitations, curbing their applicability to cybersecurity datasets.

Our work is most similar to TGAN [7] and CTGAN [8], two architectures designed for generating general-purpose tabular data. TGAN trains a recurrent neural network on tabular data in order to generate synthetic data column-by-column. CTGAN utilizes fully connected layers with residual connections in order to generate rows of tabular data. Both models are equipped to handle both categorical and numerical fields, and they utilize effective preprocessing techniques to learn and model multimodal distributions. A full comparison of our approach with TGAN and CTGAN is included in section IV.C.

C. Challenges for Cybersecurity Data Generation

Cybersecurity datasets differ from many other datasets in key ways that can pose a challenge when using GANs to generate cybersecurity data. First, cybersecurity records are often highly structured, and it can be difficult to enforce this structure using a GAN. For example, a firewall record contains a series of ordered fields, with each field containing a value of a specific type. Second, field values within records are often constrained depending on the type of the field. For example, an IPv4 address contains four octets where each octet must be within 0–255. Similarly, port numbers must be an integer within 1–65535. GANs for cybersecurity data generation must reliably enforce these constraints to generate legitimate synthetic data. Additionally, most cybersecurity datasets contain numerical and categorical fields, which complicates the GAN architecture because each type must be handled differently during the data generation process. Cybersecurity datasets often have records that are temporally dependent on each other, which are not easy to capture using simple techniques. For example, network data consists of requests and responses which are related across time. Finally, cybersecurity data such as firewall records can be quite repetitive as they capture reoccurring communication patterns over the network. This inherent repetitiveness increases the GAN's propensity for mode collapse, a common issue in GAN training where the generator becomes trapped in local minima and continuously outputs a small number of highly accurate samples that successfully fool the discriminator. Although we would like to maintain some degree of this repetition to accurately model the dataset's underlying distribution and structure, excessive repetition reduces the applicability of generated data by oversimplifying the network structure found in cybersecurity data.

III. CyberGAN: Generating High-fidelity Cybersecurity Datasets

This section first discusses our results from applying several existing GAN architectures to the simpler problem of generating IP addresses. We then extend our findings to more complex datasets and describe the details of our CyberGAN architecture.

A. Preliminary Exploration: Generating IP Addresses

To explore different strategies for generating cybersecurity data, we began by adapting and evaluating various existing architectures. We simplified the problem during this initial exploration by focusing on one common field in cybersecurity datasets: IPv4 addresses. IPv4 addresses are a good starting point because they are structured and have constraints that mirror other commonly found fields in cybersecurity datasets. IPv4 addresses consist of four eight-bit octets that range in value from 0 to 255. Any GAN architecture for generating IP addresses must maintain the structure and constraints to ensure the validity of the generated data. We investigate three GAN architectures for generating IP addresses.

PassGAN PassGAN [2] is a deep learning approach for password generation. The model utilizes the WGAN-GP framework [11] with one-dimensional convolutional layers and residual blocks to learn the underlying distribution of the training data. Initially, we attempted to treat IP address generation as a text-based problem similar to password generation. PassGAN's character-level generation and one-dimensional convolutional layers offered the potential for the network to learn the localized octet structure within the IP addresses. After training the PassGAN architecture on randomized IP addresses, approximately 96% of the generated IP addresses retained the correct structure. However, our results indicated that the model was not actually learning the IP address structure, but rather memorizing some of the training samples it had seen.

RNN WGAN RNN WGAN [3] is a deep learning architecture designed for text generation. It combines gated recurrent units (GRUs) with a GAN framework to generate text at the character level. Additionally, RNN WGAN overcomes the non-differentiability of discrete character generation via the Gumbel softmax activation function [15]. The recurrent nature of this architecture seemed promising for learning the IP addresses' octet structure. Although the model generated 100% valid samples, the samples contained repetition both within and across octets. Examples of generated IP addresses include 33.111.111.111, 44.44.44.44, 99.99.99.239, and 133.144.144.148. We believe that this repetition originated from selecting the most probable character at each step during sample generation. Ultimately, the generated samples were not useful because they did not reflect the underlying distribution of the training data.

ScratchGAN ScratchGAN [4] is a deep learning architecture for text generation that uses a recurrent generator and discriminator, discriminator regularization, word embeddings, and a dense reward structure to train a language GAN without pre-training. This model seemed promising to retain the benefits of a recurrent structure while mitigating the repetition we faced with the RNN WGAN. However, only 76% of the generated IP addresses were valid. Although the resulting samples were varied compared to the previous two approaches, the highly regimented structure of the IP addresses was perturbed.

In summary, language-based GANs are not ideal for structured data generation for two main reasons. First, the data's structure can result in repetitive outputs, as seen with the RNN WGAN. Second, most language models introduce variety into generated outputs in order to construct novel sentences. However, this variety is detrimental to highly structured data that relies on uniformity during the generation process.

B. CyberGAN Architecture

In this section, we first present the key insights from our preliminary exploration, followed by the details of our GAN architecture.

Fundamental Insight As discussed in the previous section, treating IP addresses as text leads to poor results. We instead experimented with treating their constituent octets as numerical data fields, which provided several concrete benefits. First, treating IP addresses as numerical fields helped to enforce both octet range and number constraints automatically. The GAN seemed to learn the structure and constraints better with this simple yet elegant trick. This idea also translates to other fields such as port numbers, bytes transferred, and duration. Second, processing inputs numerically allows for smoother integration of additional data types, such as categorical variables. Overall, this results in a simpler GAN architecture, which significantly decreases training time, improves the fidelity of generated data, and increases applicability to different datasets.

Data Preprocessing Data preprocessing is essential for training with complex, multimodal cybersecurity datasets. It is a crucial first step to prepare the data for ingestion by our architecture. For example, firewall records contain numerical

and categorical records, and these fields must be preprocessed accordingly. In our approach, categorical fields are one-hot encoded and concatenated to the overall input vector. We initially experimented with simple min-max normalization for numerical fields where all values were linearly rescaled to the range [-1, 1]. While this method standardized the data, it also contributed to vanishing gradients and mode collapse in GANs. The generated fields converged to a small range of values that did not accurately reflect the entire dataset's overall distribution. To more accurately reflect the multimodal nature of many cybersecurity datasets, we incorporated the mode-specific normalization used by Xu et al. [8]. This style of normalization uses a variational Gaussian mixture model (VGM) to more accurately scale numerical fields based on their corresponding mode. Numerical fields have a two-part representation: a scalar value in addition to a one-hot vector representing their mode.

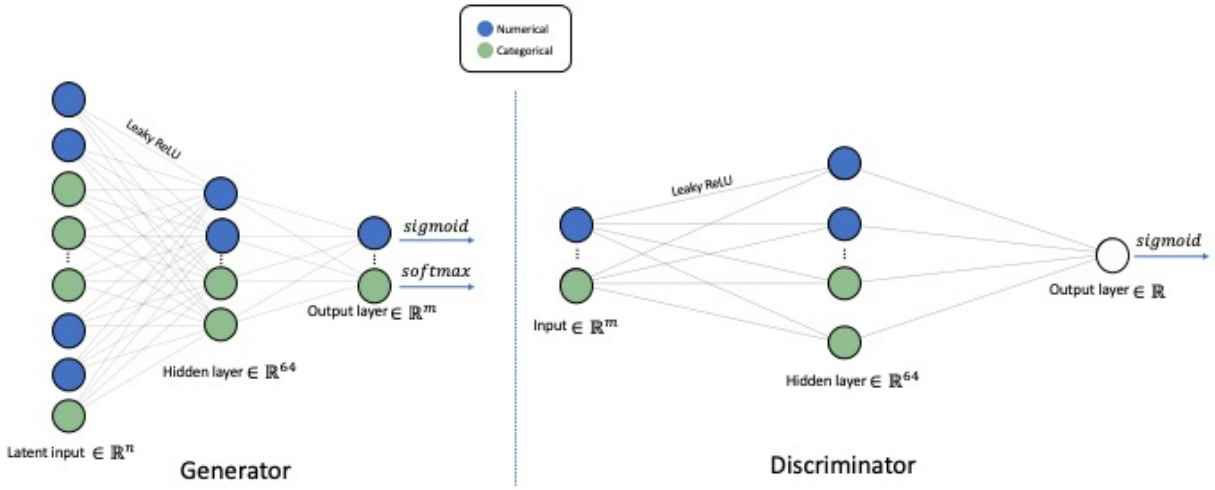


Fig. 1 The CyberGAN architecture. The generator consists of an input layer of size n, a hidden layer of size 64, and an output layer of size m, which depends on the size of the training data. Sigmoid and softmax activations are used for numerical and categorical data, respectively. The discriminator has an input layer of size m, a hidden layer of size 64, and a scalar output. The sigmoid activation function converts this scalar to the probability of the given sample being genuine. Both the generator and discriminator are fully connected with a leaky ReLU activation on the hidden layer.

Network Architecture The CyberGAN architecture, as shown in Figure 1 consists of two networks: a *generator* and a *discriminator*. Our model is based on the Wasserstein GAN with gradient penalty (WGAN-GP) [11]. Vanilla GANs notoriously suffer from training instability and mode collapse, while the WGAN-GP architecture mitigates these issues by utilizing the Earth-Mover (EM) or Wasserstein-1 distance [16]. This structure also aids convergence by penalizing the norm of the gradient of the discriminator loss with respect to its input.

CyberGAN contains a modified WGAN-GP **generator** with fully connected layers, which allows the model to learn relationships between all the fields in a record. The input to the generator is a latent vector of length n sampled from a Gaussian prior. This is followed by a single hidden layer of size 64 with leaky ReLU activation. Finally, different activation functions are applied to the output based on the type of field. The sigmoid function is applied to numerical fields, while the Gumbel softmax activation [15] is applied to categorical values. The output is a vector of length m, where m depends on the fields in the dataset. The output is then post-processed according to the normalization procedure described by Xu et al. [8].

The **discriminator** in CyberGAN is also modified from the WGAN-GP architecture. It uses fully connected layers to incorporate the relationships between and among all of the data fields. The gradient penalty is still applied to the discriminator loss as usual. The input to the discriminator is a real or generated vector of length m, where m depends on the training dataset. This input layer is followed by a singular hidden layer of size 64 with a leaky ReLU activation function. The sigmoid activation function is then applied to the output, resulting in a scalar representing the

probability that the generated sample is genuine. The relatively simple and small structure of both the generator and the discriminator has a regularizing effect and helps the model learn the training data's underlying structure without overfitting or memorizing input data.

Hyperparameter Optimization and Loss Function We use the WGAN-GP loss function described by Gulrajani et al. [11]. We also extended the loss function with a custom function to aid convergence when generating IP addresses within the desired subnet. The absolute difference between each generated and desired octet is added to the generator loss. If desired octets are not specified, we default to 0 octet loss. We also use the Adam optimizer [17] to minimize training loss. The learning rate is set to 10^{-4}. The β_1 and β_2 coefficients are set to 0.9 and 0.99, respectively, as recommended by Kingma et al. [17].

IV. Evaluation and Results

In this section, we discuss our methodology and metrics for rigorously evaluating our CyberGAN architecture. We use a combination of evaluation techniques to assess the fidelity and usefulness of the generated data. First, in Section IV.A, we evaluate the fidelity of the generated data using two metrics, which measure how well the data capture the underlying distribution. Then, in Section IV.B, we assess the suitability of the generated data for machine learning applications by testing the performance of a classifier on generated datasets. We also evaluate our GAN's performance over three different datasets: a firewall dataset, a credit card fraud dataset, and a botnet dataset. Finally, in Section IV.C, we perform a thorough comparison of our architecture to the CTGAN architecture [8].

A. Evaluating the Fidelity of Generated Data

Our objective in this evaluation is to measure the fidelity of the data generated by our GAN. We do so using two metrics: *network structure* and *Kullback-Leibler (KL) divergence*. Network structure provides us a good indication of how well the generated data captures network relationships such as connectivity, while the KL-divergence provides a reasonable idea of the statistical disparity between the two distributions. Our evaluations are performed over a publicly available firewall dataset captured from a real network *. The firewall dataset consists of network connection records between several different IP addresses. Each record contains the following key fields: time, source IP address, destination IP address, source port, destination port, protocol, bytes transferred, duration, and protocol flags. Each record captures the communication between two IP addresses on the network at a point in time.

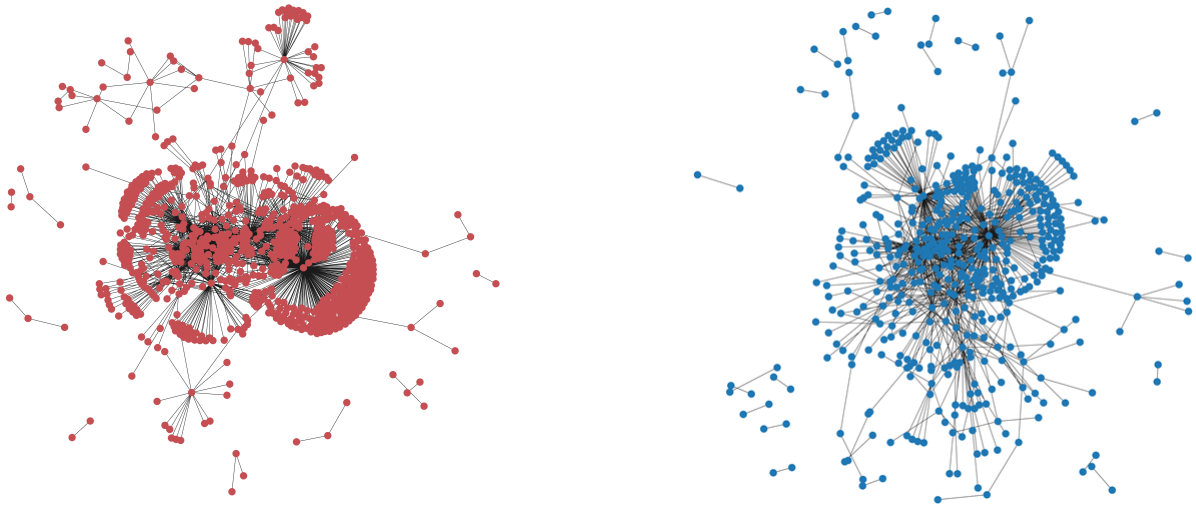
Network Structure of Generated Data The *network structure* captures the connectivity information inherent in the firewall data. Thus, comparing the network structure of the generated data to the real dataset indicates how well the GAN captures the underlying distribution of connectivity in the real dataset.

We evaluate the network structure as follows. We first train our GAN using the entire firewall dataset and then generate 10,000 firewall records using the trained network. Then, we randomly chose 10,000 records from the original firewall dataset. We compute the following network characteristics on both the original and generated datasets: number of nodes, number of edges, number of connected components, and network density. Each node is an IP address, and each edge represents at least a single communication between two IP addresses. Network density is defined as the ratio of actual edges over the possible edges in the network. A visual of the generated networks is shown in Figure 2.

The sample of the original dataset with 10,000 records contains 887 nodes, 1630 edges, and 14 connected components, resulting in a network density of 0.00415. We see that the generated data displays a similar network distribution to the original data. The graph contains 514 nodes and 943 edges with 25 connected components and a network density of 0.00715. Since the original and generated data have similar structures and network densities, we expect that the data generated by our GAN would have enough real-world characteristics to be useful for training machine learning models for cybersecurity applications.

Kullback-Leibler Divergence We next employed Kullback-Leibler (KL) Divergence to examine the difference between the distributions of the original and generated data. KL Divergence can be interpreted as a measure of the bits of information lost when using the generated data as an approximation for the real data. The average KL divergence across all fields was found to be *0.377*, highlighting the similarity between the distributions of the real and generated firewall records. This metric is not necessarily indicative of the generated data's performance and quality when used for machine learning applications. However, the KL divergence provides a good preliminary indication that our generated

*The firewall dataset is available at http://www.secrepo.com/Security-Data-Analysis/Lab_1/conn.log.zip.



(a) Original firewall data (10,000 records)

(b) Generated firewall data (10,000 records)

Fig. 2 The network structure of the original and generated firewall firewall data. Each node represents an IP address, while an edge between nodes means that those two IP addresses communicated at least once within the dataset. The network structures of the generated and real firewall data appear visually similar.

data improves upon randomly created datasets because it captures the underlying structure and relationships among data fields, which is critical for properly analyzing and understanding cybersecurity data.

B. Evaluating Usefulness of Generated Data for Machine Learning Applications

A primary objective of this work is to generate high-fidelity, shareable synthetic datasets that are useful for machine learning applications in cybersecurity. The generated data must perform similarly to real data when used to train machine learning models. One way to measure this usefulness is to test the performance of a machine learning (ML) application, such as a classifier, on both real and generated data. If the classifier can perform similarly on the generated and real datasets, we can surmise that the generated data contains enough real-world characteristics to train ML models. We do not expect our data to replace real-world datasets, but our approach can help augment or supplement existing datasets, especially when available data is sparse, as is often the case with cybersecurity datasets. We evaluate the usefulness of our approach using two different datasets: a labeled CTU-13 botnet dataset [18], and a credit card fraud detection dataset. The use of different datasets also demonstrates the applicability of our architecture to diverse cybersecurity datasets.

Evaluation of classifier performance using the CTU-13 firewall dataset The CTU-13 dataset is a labeled, publicly available dataset consisting of a mixture of normal and malicious network activity [18]. Different types of botnets are used to generate malicious network activity. We use a classifier based on the xgboost algorithm [19] to classify network activity as either malicious or benign.

In the following discussion, we refer to the CTU-13 dataset as the *real dataset*, and the GAN generated dataset as the *fake dataset*. Our approach to determining how well our generated data performs on the classification task is as follows: (1) We choose one scenario from the CTU-13 dataset and use that as the real dataset. (2) We train our GAN on the real dataset and generate a fake dataset. Approximately 1% of the original data is anomalous, so we maintained this proportion when generating data using our GAN. (3) We now evaluate the accuracy, precision, and recall of the xgboost classifier under four scenarios: train the classifier on the real data and test on real data, train on real data and test on fake data, train on fake data and test on real data, and train on fake data and test on fake data. For all the above scenarios, we use the default hyperparameters for the xgboost classifier.

As seen in Table 1, our GAN model generates fake data that functions equivalently to real data. When training the classifier on fake data and evaluating it with real data, there is a slight decrease in accuracy, precision, and recall, but the generated data still maintains much of the underlying structure that the xgboost classifier learns with the real dataset.

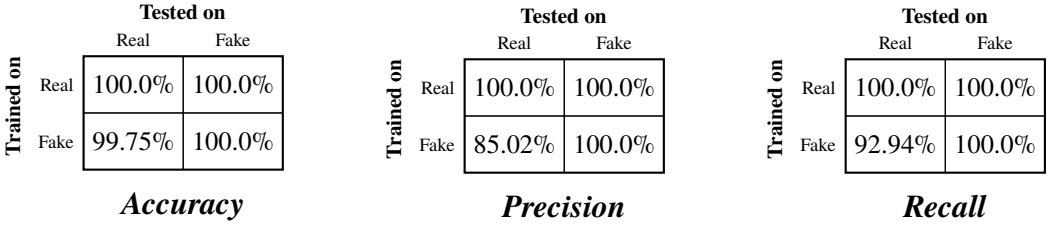


Table 1 Results of the default xgboost classifier on the CTU-13 firewall dataset. Within each category, 80% of the samples were used for training and 20% were used for testing. The generated dataset performs similarly to the real dataset when evaluating the accuracy, precision, and recall of a classifier.

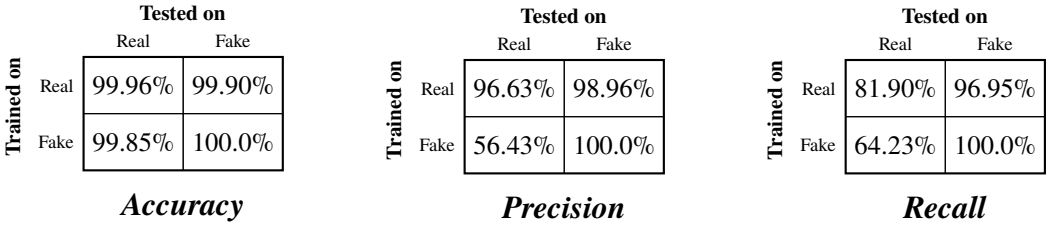


Table 2 Results of the default xgboost classifier on the credit card fraud dataset. Within each category, 80% of the samples were used for training and 20% were used for testing. Despite the low number of fraudulent samples, the fake dataset performs fairly well.

Evaluation of classifier performance with the credit card fraud detection dataset To test our architecture's applicability to machine learning applications in fields outside of cybersecurity, we utilized a well-known credit card fraud detection dataset [20–27]. The dataset consists of records representing credit card transactions, each of which is labeled normal or fraudulent. Only about 0.25% of the credit card records are fraudulent, so we maintained this proportion when generating fake data using our GAN.

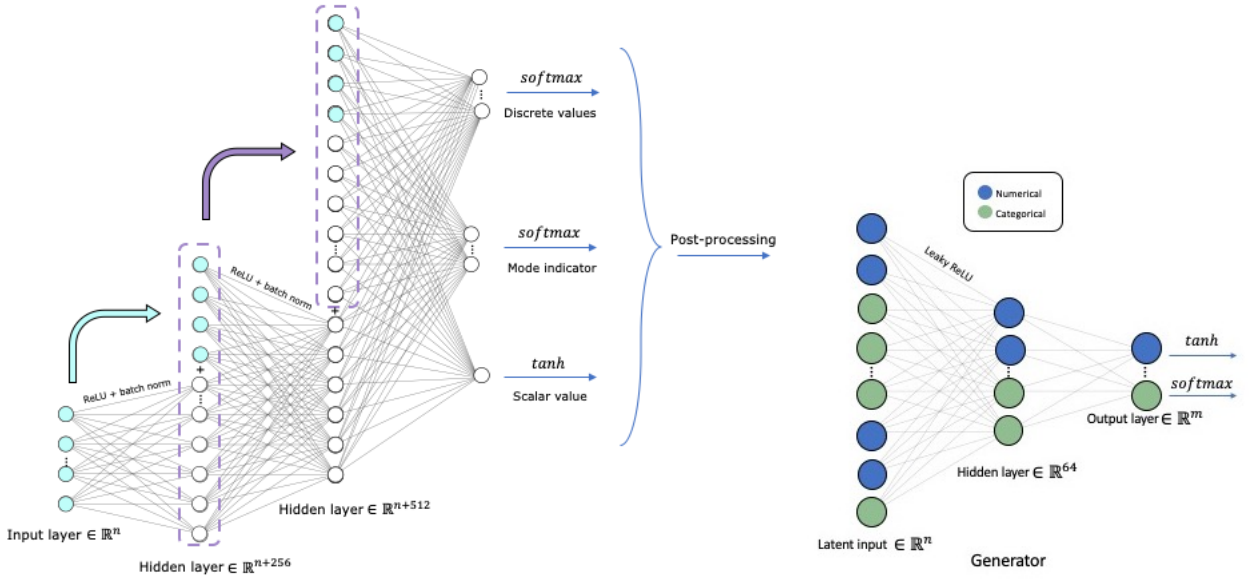
We repeated the same evaluation procedure as outlined with the CTU-13 dataset. We generate fake data using the real credit card dataset and then use the two datasets to train and test xgboost classifiers. Table 2 summarizes the results of our evaluation. As seen in the table, our GAN architecture generates fake data that is highly accurate when used to train classifiers. Although precision and recall are lower in some instances, the generated dataset mirrors the performance of the real data fairly well. In addition, low precision and recall may be at least partially attributed to the low number of fraudulent samples in both the real and generated datasets.

C. Evaluation Against the CTGAN Architecture

In this section, we perform a thorough comparison of our CyberGAN architecture with CTGAN architecture [8]. We compare the data preprocessing techniques, the architecture, and the runtimes.

The CyberGAN architecture, similar to the CTGAN architecture, incorporates preprocessing techniques using a Variational Gaussian Mixture Model (VGM) to effectively preprocess multimodal distributions while avoiding vanishing gradients and mode collapse. Similarly, both architectures use fully connected layers to learn relationships across fields and apply different activation functions to the output to account for categorical and discrete data. A comparison of the architectures is shown in Figure 3. While CTGAN is a framework for general tabular data, our architecture tackles specific challenges when generating cybersecurity datasets with applications for evaluating machine learning techniques. For instance, when generating firewall records with CTGAN, many of the generated records were negative or had values that were not reasonable in the context of cybersecurity fields such as IP addresses and ports. To generate non-negative records within a more reasonable range, we use a simpler architecture in addition to a sigmoid activation instead of tanh on the output of our generator.

CTGAN also employs other techniques to prevent mode collapse and encourage variety in the generated data, such as sample packing [28], training by sampling [8], residual connections, and batch normalization [29]. However, since



(a) CTGAN (b) CyberGAN

Fig. 3 A comparison of the CTGAN and CyberGAN generator architectures. The CTGAN generator incorporates residual connections (represented by arrows) in addition to ReLU activation and batch normalization. On the other hand, the CyberGAN generator has fully connected layers with a leaky ReLU activation on the hidden layer.

we prioritize maintaining the structure of the original dataset, our architecture is much simpler and allows the underlying data distribution to drive sample generation. For example, since firewall records are inherently repetitive as the same machines frequently communicate during a given time period, excessive diversity in generated data would disturb the structure of generated data and make it less realistic. Packing techniques used in CTGAN proved to be detrimental to firewall record generation as they reduced the connectivity in the network structure of the generated data.

CTGAN's complexity also contributes significantly to the training time. We evaluated the performance of CyberGAN and CTGAN architectures on the firewall dataset. Our model finished training in just under 2 hours (including preprocessing), and the CTGAN architecture took more than 24 hours to complete ten epochs.

V. Discussion

In this section, we summarise our key results, discuss the current limitations of our architecture, and outline the next steps for this research.

A. Summary of Results

The different evaluations discussed in Section IV demonstrate that CyberGAN is able to generate datasets suitable for cybersecurity applications. CyberGAN's performance on cybersecurity datasets is better than the performance of other GAN architectures, such as the RNN WGAN and CTGAN. As discussed in previous sections, cybersecurity data has inherent characteristics that lead to complications with traditional architectures. For example, firewall records are highly structured, and the fields within each record have constraints that are not handled well by existing GAN architectures.

We evaluated the fidelity and usefulness of data generated by CyberGAN. As discussed in Section IV.A, CyberGAN was able to generate firewall data that resembled a network structure very close to the real dataset. Further, the KL-divergence metric confirmed that the generated dataset could capture the underlying distribution of the real dataset. Then, as discussed in Section IV.B, CyberGAN was able to generate data that performed similarly to a real dataset on a machine learning classification task. We do not expect our generated datasets to replace real datasets, but our approach can help augment existing datasets, especially when available data is sparse, as is often the case with cybersecurity datasets. For instance, ML-based anomaly detectors offer the promise of detecting novel cybersecurity attacks. However, the lack of attack or anomaly samples often makes it hard to effectively train ML-based anomaly detectors. CyberGAN can augment existing datasets by generating fake anomalous records that still maintain the underlying characteristics

and patterns of real anomalous data.

Lastly, we also demonstrated that CyberGAN is easily adaptable to a variety of datasets. In this work, we demonstrated the application of CyberGAN to three different datasets: a firewall dataset, the CTU-13 botnet dataset, and the credit card fraud detection dataset.

B. Limitations

Data generated with our model is primarily intended to be highly shareable to alleviate privacy and security concerns associated with sharing traditional cybersecurity data. To that end, our model is designed to generate high-fidelity synthetic data that functions similarly to real data in machine learning applications but is not a carbon copy of the original dataset. However, since GANs generally learn the underlying structure and distribution of the training data, we cannot guarantee that the synthetic dataset does not contain statistical attributes or fields that would allow a user to recreate portions of the original data. We performed preliminary investigations into the similarity of the generated data to the real dataset. For example, we found that real IP addresses (found in the training dataset) were not present in a sample of 10,000 firewall records generated by our GAN. This result is encouraging but is not proof that CyberGAN guarantees privacy. In the future, we plan to investigate techniques that guarantee differential privacy [30].

C. Future Work

There are several opportunities for improving upon our current work in domains such as privacy and generalizability. For instance, the data currently generated by our model is unconstrained based on the input latent vector and the model's training. In the future, we may extend this framework using conditional GANs [31]. By providing a starting condition to the GAN, we can enforce a myriad of constraints on the generated data, such as the range of octets in generated IP addresses. These constraints are useful because they allow for a greater specificity when generating data, which can aid in individualized data creation for particular cybersecurity machine learning tasks. Similarly, labeled data must currently be generated one category at a time, i.e., the model must be separately trained on anomalous and normal records to generate the appropriate category. Incorporating a conditional GAN into our framework could provide the structure necessary to generate a mixture of malicious and normal records by specifying the desired category as one of the conditions inputted to the GAN.

Further, cybersecurity datasets often contain records that are highly dependent on each other across time. For example, in addition to relationships between the fields, firewall records exhibit temporal relationships such as frequency and periodicity of communication between machines in the network. Our future work includes adding a temporal aspect to data generation with our architecture, potentially through recurrent neural networks (RNNs). Yoon and Jarrett et al. [32] have previously done work on applying GANs to time-series data, and we plan to incorporate such techniques into our architecture.

VI. Conclusion

The lack of high-fidelity, shareable datasets impedes cybersecurity research. We introduced CyberGAN, a GAN-based architecture for generating high-fidelity, shareable cybersecurity datasets. Comparisons to existing GAN architectures show that our architecture is simple, fast, and accurate in generating different datasets. We rigorously evaluated the fidelity and usefulness of our generated data using multiple datasets. First, we measured fidelity using network characteristics and KL-divergence in a generated firewall dataset, and both metrics indicate that CyberGAN was able to capture the distribution in the training dataset sufficiently. Then, we evaluated the usefulness of our GAN-generated dataset for machine learning applications by assessing the performance of a machine learning classifier (xgboost) on our generated data. Our results demonstrated that the classifier performed similarly on the generated and real datasets. We do not expect the generated data by our GAN to replace real-world datasets, but our approach can help augment or supplement existing datasets, especially when available data is sparse, as is often the case with cybersecurity datasets. Overall, our approach seems to be a step in the right direction towards generating high-fidelity datasets that possess real-world characteristics. We hope that this paper serves as a catalyst for further research in this area.

VII. Acknowledgements

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] Hitaj, B., Gasti, P., Ateniese, G., and Perez-Cruz, F., "PassGAN: A deep learning approach for password guessing," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 11464 LNCS, 2019, pp. 217–237. https://doi.org/10.1007/978-3-030-21568-2_11.

[3] Press, O., Bar, A., Bogin, B., Berant, J., and Wolf, L., "Language generation with recurrent generative adversarial networks without pre-training," *arXiv preprint arXiv:1706.01399*, 2017.

[4] de Masson d'Autume, C., Mohamed, S., Rosca, M., and Rae, J., "Training language gans from scratch," *Advances in Neural Information Processing Systems*, 2019, pp. 4300–4311.

[5] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N., "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5907–5915.

[6] Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., and He, X., "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1316–1324.

[7] Xu, L., and Veeramachaneni, K., "Synthesizing tabular data using generative adversarial networks," *arXiv preprint arXiv:1811.11264*, 2018.

[8] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K., "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, 2019, pp. 7335–7345.

[9] Yahi, A., Vanguri, R., Elhadad, N., and Tatonetti, N. P., "Generative adversarial networks for electronic health records: A framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories," *arXiv preprint arXiv:1712.00164*, 2017.

[10] Fekri, M. N., Ghosh, A. M., and Grolinger, K., "Generating energy data for machine learning with recurrent generative adversarial networks," *Energies*, Vol. 13, No. 1, 2020, p. 130.

[11] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C., "Improved training of wasserstein gans," *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[12] Reiter, J. P., "Using CART to generate partially synthetic public use microdata," *Journal of Official Statistics*, Vol. 21, No. 3, 2005, p. 441.

[13] Aviñó, L., Ruffini, M., and Gavaldà, R., "Generating synthetic but plausible healthcare record datasets," *arXiv preprint arXiv:1807.01514*, 2018.

[14] Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., and Xiao, X., "Privbayes: Private data release via bayesian networks," *ACM Transactions on Database Systems (TODS)*, Vol. 42, No. 4, 2017, pp. 1–41.

[15] Jang, E., Gu, S., and Poole, B., "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[16] Arjovsky, M., Chintala, S., and Bottou, L., "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[17] Kingma, D. P., and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] Garcia, S., Grill, M., Stiborek, J., and Zunino, A., "An empirical comparison of botnet detection methods," *computers & security*, Vol. 45, 2014, pp. 100–123.

[19] Chen, T., and Guestrin, C., "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, 2016, pp. 785–794. https://doi.org/10.1145/2939672.2939785, URL http://doi.acm.org/10.1145/2939672.2939785.

[20] Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G., "Calibrating probability with undersampling for unbalanced classification," *2015 IEEE Symposium Series on Computational Intelligence*, IEEE, 2015, pp. 159–166.

[21] Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., and Bontempi, G., "Learned lessons in credit card fraud detection from a practitioner perspective," *Expert systems with applications*, Vol. 41, No. 10, 2014, pp. 4915–4928.

[22] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G., "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE transactions on neural networks and learning systems*, Vol. 29, No. 8, 2017, pp. 3784–3797.

[23] Dal Pozzolo, A., "Adaptive machine learning for credit card fraud detection," 2015.

[24] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., and Bontempi, G., "Scarff: a scalable framework for streaming credit card fraud detection with spark," *Information fusion*, Vol. 41, 2018, pp. 182–194.

[25] Carcillo, F., Le Borgne, Y.-A., Caelen, O., and Bontempi, G., "Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization," *International Journal of Data Science and Analytics*, Vol. 5, No. 4, 2018, pp. 285–300.

[26] Lebichot, B., Le Borgne, Y.-A., He-Guelton, L., Oblé, F., and Bontempi, G., "Deep-learning domain adaptation techniques for credit cards fraud detection," *INNS Big Data and Deep Learning conference*, Springer, 2019, pp. 78–88.

[27] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., and Bontempi, G., "Combining unsupervised and supervised learning in credit card fraud detection," *Information Sciences*, 2019.

[28] Lin, Z., Khetan, A., Fanti, G., and Oh, S., "Pacgan: The power of two samples in generative adversarial networks," *Advances in neural information processing systems*, 2018, pp. 1498–1507.

[29] Ioffe, S., and Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[30] Jordon, J., Yoon, J., and van der Schaar, M., "PATE-GAN: Generating synthetic data with differential privacy guarantees," *International Conference on Learning Representations*, 2018.

[31] Mirza, M., and Osindero, S., "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[32] Yoon, J., Jarrett, D., and van der Schaar, M., "Time-series generative adversarial networks," *Advances in Neural Information Processing Systems*, 2019, pp. 5508–5518.